|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实验名称** |  | | **实验考核成绩** |  |
| **实验者** | **设计者、验证者** | |
| **实验地点** |  | |
| **实验目的及要求** | | | | |
| **一、实验目的**  能够熟练使用My Sql，能熟练掌握和操作SQL语句。  **一、实验要求**   1. 熟悉My Sql的数据库开发环境； 2. 掌握创建、删除和使用数据库的方法； 3. 掌握创建、删除、修改等对表的操作； 4. 掌握数据库中用户的创建和授权； | | | | |
| **实验内容及操作流程** | | | | |
| 1. **实验内容** 2. 在MySql中，利用命令模式分别完成以下内容：   **请把当时发给大家的要求粘贴至此处。**   1. **操作流程**   **请把完成的操作步骤粘贴至此处，注意每次的截图过程需要进行描述。**  **1.对系统进行需求分析，识别出系统的数据项及数据结构，并完成系**  **统的概念模型;**  学生表（students）：  - student\_id（学生ID，主键）  - name（学生姓名）  - grade（年级）  - class（班级）  - major\_id（专业ID，外键引用专业表中的major\_id）  专业表（majors）：  - major\_id（专业ID，主键）  - major\_name（专业名称）  课程表（courses）：  - course\_id（课程ID，主键）  - course\_name（课程名称）  - credit（学分）  学生成绩表（scores）：  - score\_id（成绩ID，主键）  - student\_id（学生ID，外键引用学生表中的student\_id）  - course\_id（课程ID，外键引用课程表中的course\_id）  - score（成绩）  **2.通过系统的概念模型完成系统的逻辑结构设计，以二维表的形式展**  **示各实体以及实体间关系，并描述各数据项的字段名称、类型以及约**  **束条件等:**  **学生表：**   |  |  |  | | --- | --- | --- | | **字段名称** | **类型** | **约束条件** | | student\_id | INT | 主键 | | name | VARCHAR(50) |  | | grade | INT |  | | class | INT |  | | major\_id | INT | 外键，参考 majors(major\_id) |   **专业表：**   |  |  |  | | --- | --- | --- | | **字段名称** | **类型** | **约束条件** | | major\_id | INT | 主键 | | major\_name | VARCHAR(50) |  |   **成绩表：**   |  |  |  | | --- | --- | --- | |  |  |  | | **字段名称** | **类型** | **约束条件** | | score\_id | INT | 主键 | | student\_id | INT | 外键，参考 students(student\_id) | | course\_id | INT | 外键，参考 courses(course\_id) | | score | FLOAT |  |   **课程表：**   |  |  |  | | --- | --- | --- | | **字段名** | **类型** | **约束条件** | | course\_id | INT | PRIMARY KEY | | course\_name | VARCHAR(50) | NOT NULL | | credit | FLOAT | NOT NULL |   **3.安装Mysql数据库，通过命令方式创建数据库(StudentManage)**  -- 创建数据库  CREATE DATABASE StudentManage;    **4.为数据库创建两个用户，一个为(Lee)具有创建、修改、删除、查询**  **的权限，另一个(Tom)只具有对表进行查询的权限;**  -- 创建用户Lee  CREATE USER 'Lee'@'localhost' IDENTIFIED BY 'password';  GRANT ALL PRIVILEGES ON StudentManage.\* TO 'Lee'@'localhost';  -- 创建用户Tom  CREATE USER 'Tom'@'localhost' IDENTIFIED BY 'password';  GRANT SELECT ON StudentManage.\* TO 'Tom'@'localhost';    **5.使用Len身份为StudentManage数据库创建表，注意主键约束、外键**  **约束和用户定义的约束**  -- 使用Lee用户身份创建表  USE StudentManage;  -- 创建学生表  CREATE TABLE students (  student\_id INT PRIMARY KEY,  name VARCHAR(50),  grade INT,  class INT,  major\_id INT,  FOREIGN KEY (major\_id) REFERENCES majors(major\_id)  );  -- 创建专业表  CREATE TABLE majors (  major\_id INT PRIMARY KEY,  major\_name VARCHAR(50)  );  -- 创建课程表  CREATE TABLE courses (  course\_id INT PRIMARY KEY,  course\_name VARCHAR(50) NOT NULL,  credit FLOAT NOT NULL  );  -- 创建学生成绩表  CREATE TABLE scores (  score\_id INT PRIMARY KEY,  student\_id INT,  course\_id INT,  score FLOAT,  FOREIGN KEY (student\_id) REFERENCES students(student\_id),  FOREIGN KEY (course\_id) REFERENCES courses(course\_id)  );    **6.使用Lee用户身份向每个表中插入至少5条数据:**  -- 使用Lee用户身份插入数据  USE StudentManage;  -- 插入学生表数据  INSERT INTO students (student\_id, name, grade, class, major\_id)  VALUES  (1, '张三', 1, 1, 1),  (2, '李四', 2, 2, 2),  (3, '王五', 3, 3, 1),  (4, '赵六', 2, 1, 3),  (5, '钱七', 3, 2, 2);  -- 插入专业表数据  INSERT INTO majors (major\_id, major\_name)  VALUES  (1, '计算机科学与技术'),  (2, '电子工程'),  (3, '化学工程'),  (4, '机械工程'),  (5, '经济学');  -- 插入课程表数据  INSERT INTO courses (course\_id, course\_name, credit)  VALUES  (1, '数据库', 3.0),  (2, '计算机网络', 3.5),  (3, '操作系统', 4.0),  (4, '物理化学', 3.5),  (5, '微观经济学', 3.0);  -- 插入学生成绩表数据  INSERT INTO scores (score\_id, student\_id, course\_id, score)  VALUES  (1, 1, 1, 85.5),  (2, 2, 3, 92.0),  (3, 3, 2, 78.5),  (4, 4, 4, 89.0),  (5, 5, 5, 95.5);      **7.支持查询学生选课信息包括学生编号、学生姓名、选修课程号、选**  **修课程名、授课教师和授课地点，为了加快数据的查询，为表建立起**  **正确的索引;**  -- 使用Lee用户身份建立索引  USE StudentManage;  -- 为学生表建立索引  CREATE INDEX idx\_student\_major\_id ON students (major\_id);    **8.支持查询各科成绩大于60分的学生学生发放毕业证书，学生各科成**  **绩平均在70分以上并且每科都在60分以上的学生才能放发学位证书，**  **查询显示的信息需要包括学号、姓名、平均分、最高分、最低分、所**  **在班级、所属专业名称(为了方便以后的查询通过视图方式对查询信**  **息进行封装)，需要支持指定年级或者学生查询;**  -- 使用Tom用户身份查询学生选课信息  USE StudentManage;  SELECT s.student\_id, s.name, c.course\_id, c.course\_name, c.teacher, c.location  FROM students AS s  JOIN courses AS c ON s.student\_id = c.student\_id;  -- 使用Tom用户身份查询符合条件的学生信息  USE StudentManage;  SELECT s.student\_id, s.name, AVG(score) AS average\_score,  MAX(score) AS max\_score, MIN(score) AS min\_score,  s.class, m.major\_name  FROM students AS s  JOIN scores AS sc ON s.student\_id = sc.student\_id  JOIN majors AS m ON s.major\_id = m.major\_id  GROUP BY s.student\_id, s.name, s.class, m.major\_name  HAVING MIN(score) >= 60;    SELECT s.student\_id, s.name, AVG(score) AS average\_score,  MAX(score) AS max\_score, MIN(score) AS min\_score,  s.class, m.major\_name  FROM students AS s  JOIN scores AS sc ON s.student\_id = sc.student\_id  JOIN majors AS m ON s.major\_id = m.major\_id  GROUP BY s.student\_id, s.name, s.class, m.major\_name  HAVING AVG(score) >= 70 AND MIN(score) >= 60;    **9.支持对专业编号的修改，所有要求专业编号修改后，同时必须更新**  **学生信息表中学生所在专业的专业号;**  -- 创建触发器  DELIMITER //  CREATE TRIGGER update\_major\_id  AFTER UPDATE ON majors  FOR EACH ROW  BEGIN  IF NEW.major\_id <> OLD.major\_id THEN  UPDATE students SET major\_id = NEW.major\_id WHERE major\_id = OLD.major\_id;  END IF;  END//  DELIMITER ;    **10.为了保障数据库的安全，需要给数据库进行定期备份，当出现问题**  **时能够即时恢复，需要编写对StudentManage数据库的备份和恢复语**  **句;**  -- 备份数据库  mysqldump -u root -p StudentManage > student\_manage\_backup.sql    -- 恢复数据库  mysql -u root -p StudentManage < student\_manage\_backup.sql | | | | |
| **总结成绩** | |  | | |
| **总结**  **（遇到的问题，如何解决以及解决方法，可自行添加问题数目）** | | | | |
| **问题一** | | | | |
| 创建外键需要注意拥有目标表的权限和目标表列要存在。 | | | | |
| **问题二** | | | | |
| 对于有外键的表，插入和删除要注意改记录有没有被使用，不然需要先断开连接，才能实现删除等操作。 | | | | |